

CS516PROJECT

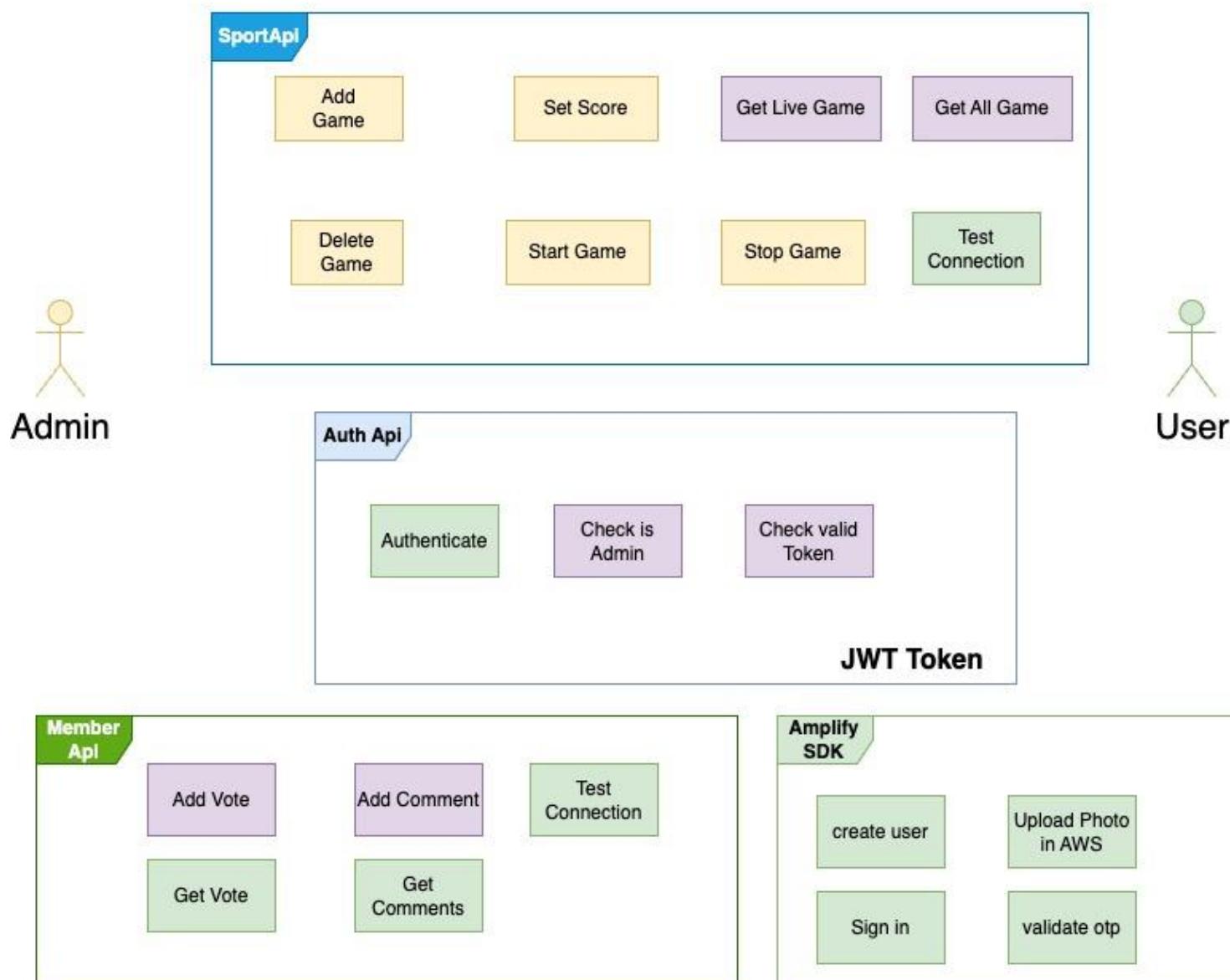
+ .
o

Team Members

Ariunzaya Munkhbat
Bayarbayasgalan Jagdal
Syed Shujat Ali Shah
Thanh Do Nguyen
Duyen Tran



APPLICATION IDEA



AWS Cloud

Microservices

Databases
(MongoDB + RDS)

S3 Storage
(Cloud-watch, images, logs)

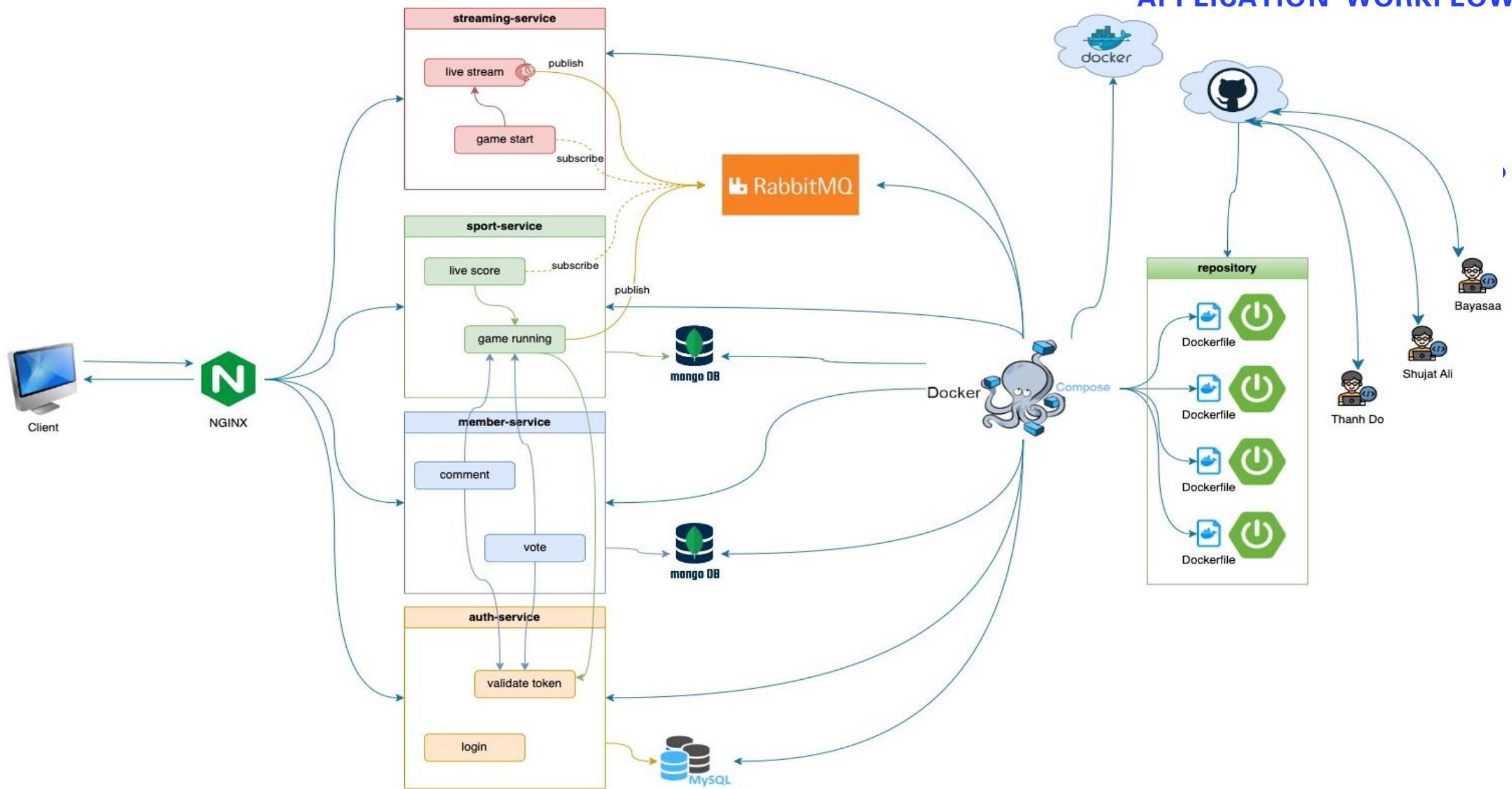
Lambda functions

Api Gateway

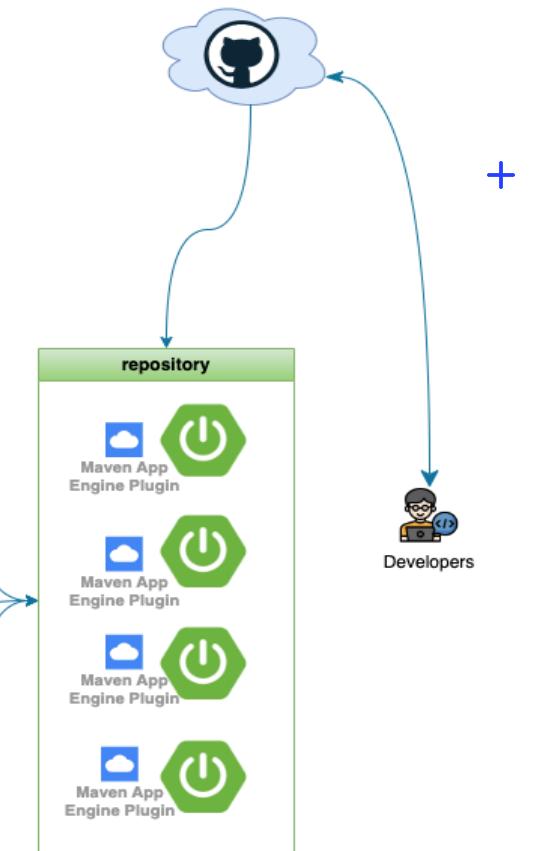
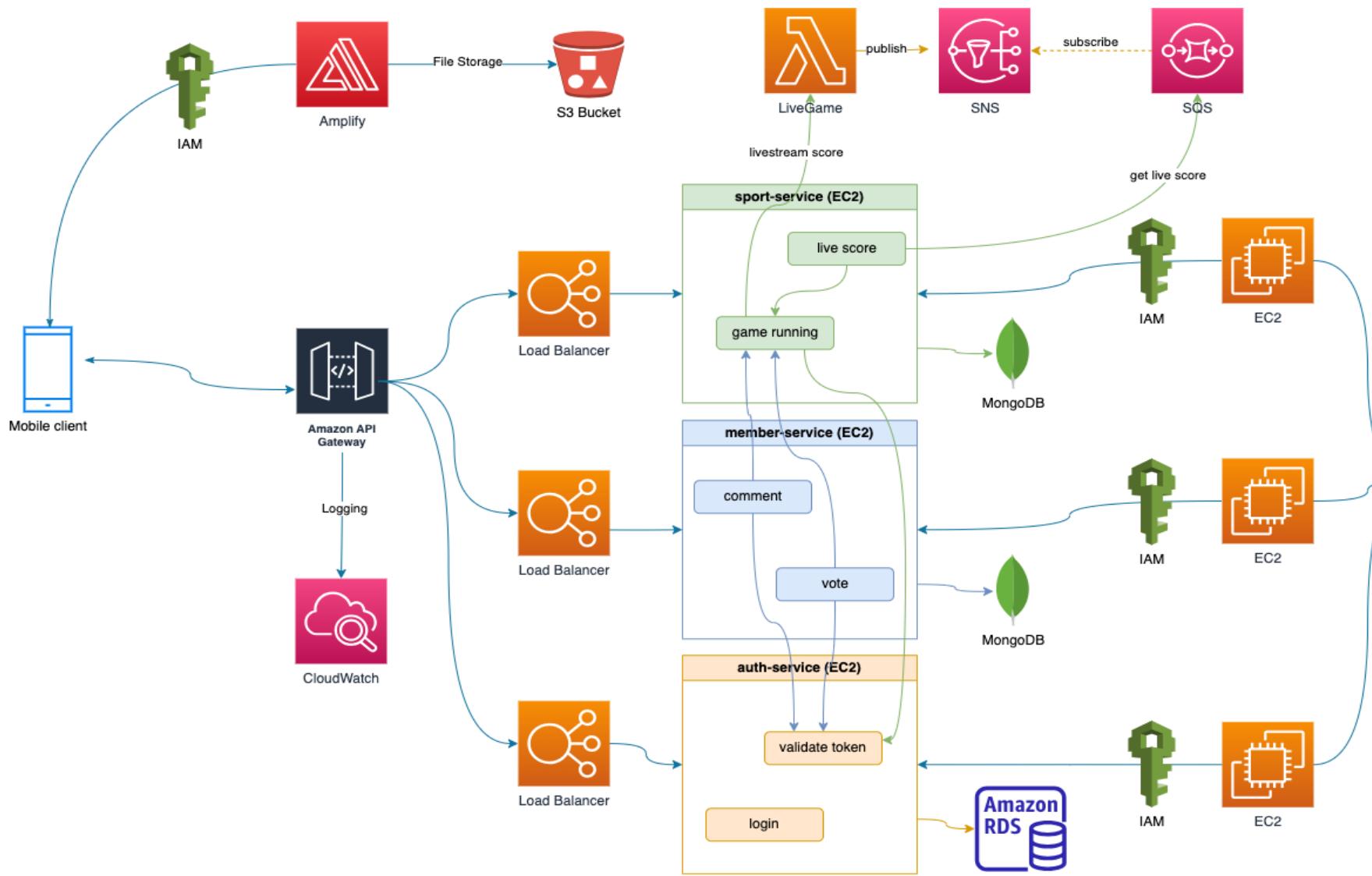
Android Mobile App Client

AWS Amplify

APPLICATION WORKFLOW



APPLICATION ON AWS



EC2 Instances

Instances (3) [Info](#)

[Connect](#) [Instance state ▾](#) [Actions ▾](#) [Launch instances](#) ▾

Find instance by attribute or tag (case-sensitive)

[Instance state = running](#) [X](#) [Clear filters](#)

<input type="checkbox"/>	Name ▾	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone ▾	Public IPv4 DNS	
<input type="checkbox"/>	Sport-Service	i-008bc914d1b36111d	✓ Running Q Q	t2.micro	✓ 2/2 checks passed	No alarms	+	us-east-1c	ec2-52-91-229-223.compute-1.amazonaws.com
<input type="checkbox"/>	AuthService	i-042fe293d5e3ebdfb	✓ Running Q Q	t2.micro	✓ 2/2 checks passed	No alarms	+	us-east-1c	ec2-44-202-164-100.compute-1.amazonaws.com
<input type="checkbox"/>	MemberService	i-0c2a7729ec5b23afc	✓ Running Q Q	t2.micro	✓ 2/2 checks passed	No alarms	+	us-east-1c	ec2-54-173-93-46.compute-1.amazonaws.com

[<](#) [1](#) [>](#) [⚙️](#)

Select an instance

[=](#) [⚙️](#) [X](#)

Running applications

```
git clone https://github.com/bayarbayasgalanj/cloud\_computing.git
```

```
Install Java
```

```
sudo apt-get update
```

```
sudo apt install openjdk-17-jre
```

```
sudo apt install openjdk-17-jdk
```

```
sudo apt install -y maven
```

```
Install Mongo
```

```
wget -qO - https://www.mongodb.org/static/pgp/server-4.2.asc | sudo apt-key add -
```

```
echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.2 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.2.list
```

```
sudo apt-get update
```

```
sudo apt install -y mongodb-org
```

```
echo "deb http://security.ubuntu.com/ubuntu focal-security main" | sudo tee /etc/apt/sources.list.d/focal-security.list
```

```
sudo apt-get update
```

```
sudo apt-get install libssl1.1
```

```
sudo apt install -y mongodb-org
```

```
sudo nano /etc/mongod.conf
```

```
net:
```

```
port: 27017
```

```
bindip: 0.0.0.0
```

```
sudo systemctl start mongod
```

```
sudo systemctl enable mongod
```

```
Start service
```

```
mvn clean -f pom.xml
```

```
mvn package -f pom.xml
```

```
java -jar target/sport-service-1.0-SNAPSHOT.jar
```

EC2 Sport service

ec2-52-91-229-223.compute-1.amazonaws.com:8081/sport/swagger-ui/index.html?configUrl=/sport/api-docs/swagger-config

Swagger
Supported by SMARTBear

/sport/api-docs Explore

OpenAPI definition v0 OAS3 /sport/api-docs

Servers http://ec2-52-91-229-223.compute-1.amazonaws.com:8081/sport - Generated server url ▾

deleteAllOthers deleteAll without live Game

DELETE /deleteAllOthers

get get just one Game by ID

GET /get

getAllGame coming the all games

GET /getAllGame

deleteAll deleteAll Game

DELETE /deleteAll

getLiveGame getting just Live Game

GET /getLiveGame

setStart It is starting game also sending to Message Queue

POST /setStart

setScore It is setting the scores

POST /setScore

delete delete just one Game by ID

DELETE /delete

addGame It is adding new game to Mongodb

POST /addGame

setStop It is stoping game also set the who is win

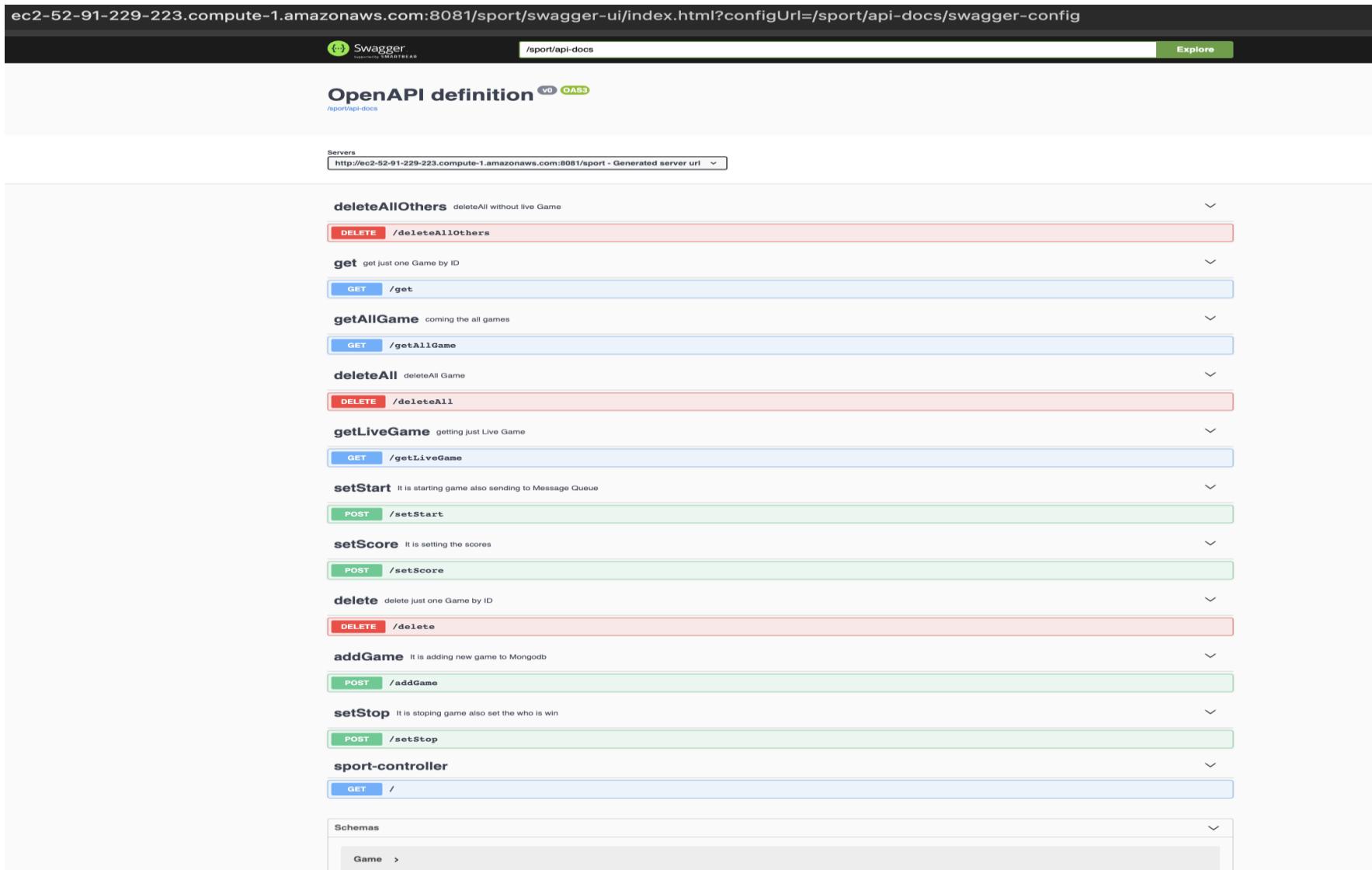
POST /setStop

sport-controller

GET /

Schemas

Game >



Lambda invoker from Sport service using AWS SDK

```
▲ nthanhdo2610
@Service
public class LambdaInvoker {

    @Value("${aws.accessKeyId}")
    private String accessKeyId;
    @Value("${aws.secretAccessKey}")
    private String secretAccessKey;
    @Value("${aws.lambda-function}")
    private String lambdaFunctionName;
    2 usages
    private LambdaClient lambdaClient;
    ▲ nthanhdo2610
    @PostConstruct
    private void init() {
        // Set up the Lambda client
        lambdaClient = LambdaClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(StaticCredentialsProvider.create(AwsBasicCredentials.create(accessKeyId, secretAccessKey)))
            .build();
    }

    3 usages ▲ nthanhdo2610
    public void sendMessage(String message){
        // Create an InvokeRequest
        InvokeRequest request = InvokeRequest.builder()
            .functionName(lambdaFunctionName)
            .payload(SdkBytes.fromUtf8String("{\"message\":\"" + message + "\"}"))
            .build();

        try {
            // Invoke the Lambda function
            InvokeResponse response = lambdaClient.invoke(request);

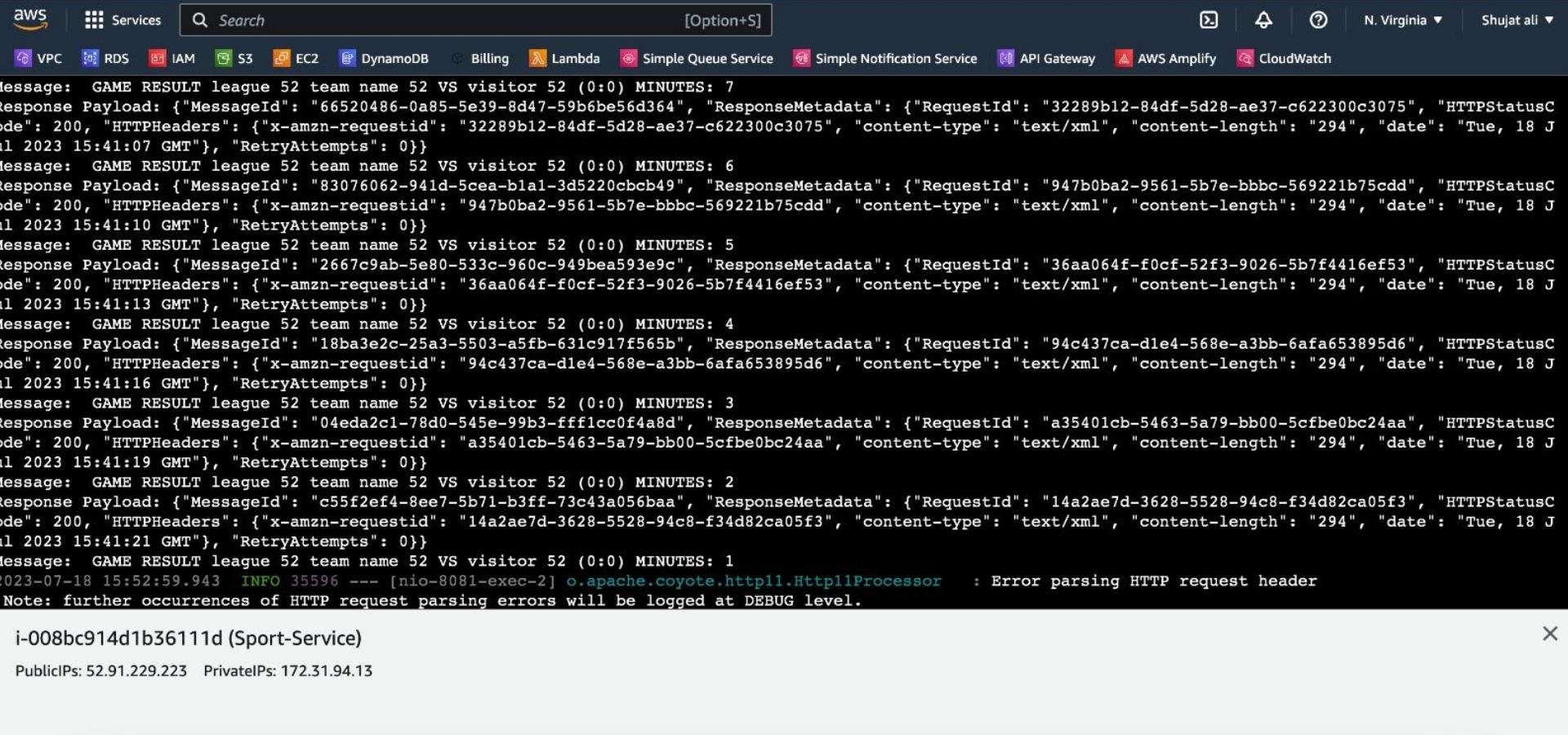
            // Get the response payload
            String payload = new String(response.payload().asByteArray(), StandardCharsets.UTF_8);
            System.out.println("Response Payload: " + payload);
        } catch (LambdaException e) {
            System.err.println("Error invoking Lambda function: " + e.getMessage());
        }
    }
}
```

SQS Listener from Sport service using AWS SDK

```
public class SQSMessageListener {  
    2 usages  
    private final Region region = Region.US_EAST_1;  
    @Value("${aws.accessKeyId}")  
    private String accessKeyId;  
    @Value("${aws.secretAccessKey}")  
    private String secretAccessKey;  
    6 usages  
    private SqsClient sqsClient;  
    5 usages  
    private static String queueUrl;  
    2 usages  
    private ReceiveMessageRequest receiveRequest;  
    ± nthanhdo2610 *  
    @PostConstruct  
    private void init() {  
        SnsClient snsClient = SnsClient.builder().region(region)  
            .credentialsProvider(StaticCredentialsProvider.create(AwsBasicCredentials.create(accessKeyId, secretAccessKey))).build();  
        sqsClient = SqsClient.builder().region(region)  
            .credentialsProvider(StaticCredentialsProvider.create(AwsBasicCredentials.create(accessKeyId, secretAccessKey))).build();  
        // Create an SNS topic  
        CreateTopicRequest createTopicRequest = CreateTopicRequest.builder().name(s: "LiveScore").build();  
        CreateTopicResponse createTopicResponse = snsClient.createTopic(createTopicRequest);  
        String topicArn = createTopicResponse.topicArn();  
        // Create two SQS queues  
        CreateQueueRequest createQueueRequest = CreateQueueRequest.builder().queueName(s: "Score").build();  
        CreateQueueResponse createQueueResponse = sqsClient.createQueue(createQueueRequest);  
        queueUrl = createQueueResponse.queueUrl();  
        // Get the ARNs of the queues  
        String queueArn = sqsClient  
            .getQueueAttributes(GetQueueAttributesRequest.builder().queueUrl(queueUrl).attributeNames(QueueAttributeName.QUEUE_ARN).build())  
            .attributes().get(QueueAttributeName.QUEUE_ARN);  
        // Subscribe the queues to the SNS topic  
        SubscribeRequest subscribeRequest1 = SubscribeRequest.builder().topicArn(topicArn).protocol(s: "sqS").endpoint(queueArn).build();  
        snsClient.subscribe(subscribeRequest1);  
        // Set the SQS queue attributes to receive messages from SNS  
        Map<QueueAttributeName, String> queueAttributes = new HashMap<>();  
        queueAttributes.put(QueueAttributeName.RECEIVE_MESSAGE_WAIT_TIME_SECONDS, "20");  
        queueAttributes.put(QueueAttributeName.VISIBILITY_TIMEOUT, "300");  
        SetQueueAttributesRequest setQueueAttributesRequest = SetQueueAttributesRequest.builder().queueUrl(queueUrl).attributes(queueAttributes).build();  
        sqsClient.setQueueAttributes(setQueueAttributesRequest);  
        receiveRequest = ReceiveMessageRequest.builder().queueUrl(queueUrl).maxNumberOfMessages(integer: 10).waitTimeSeconds(integer: 20).build();  
    }  
}
```

```
± nthanhdo2610 *  
@Scheduled(fixedDelay = 1000)  
public void receiveAndProcessMessages() {  
    ReceiveMessageResponse receiveResponse = sqsClient.receiveMessage(receiveRequest);  
    for (Message message : receiveResponse.messages()) {  
        try {  
            JSONObject obj = new JSONObject(message.body());  
            System.out.println("Message: " + obj.getString(name: "Message"));  
        } catch (JSONException e) {  
            System.out.println("JSONException Message: " + e.getMessage());  
        }  
        deleteMessage(queueUrl, message.receiptHandle());  
    }  
}  
  
1 usage ± nthanhdo2610  
private void deleteMessage(String queueUrl, String receiptHandle) {  
    DeleteMessageRequest deleteRequest = DeleteMessageRequest.builder()  
        .queueUrl(queueUrl)  
        .receiptHandle(receiptHandle)  
        .build();  
  
    sqsClient.deleteMessage(deleteRequest);  
}
```

SQS Listener from Sport service using AWS SDK



The screenshot shows the AWS CloudWatch Logs interface with the following log entries:

```
Message: GAME RESULT league 52 team name 52 VS visitor 52 (0:0) MINUTES: 7
Response Payload: {"MessageId": "66520486-0a85-5e39-8d47-59b6be56d364", "ResponseMetadata": {"RequestId": "32289b12-84df-5d28-ae37-c622300c3075", "HTTPStatusCode": 200, "HTTPHeaders": {"x-amzn-requestid": "32289b12-84df-5d28-ae37-c622300c3075", "content-type": "text/xml", "content-length": "294", "date": "Tue, 18 Jul 2023 15:41:07 GMT"}, "RetryAttempts": 0}}
Message: GAME RESULT league 52 team name 52 VS visitor 52 (0:0) MINUTES: 6
Response Payload: {"MessageId": "83076062-941d-5cea-b1a1-3d5220cbcba9", "ResponseMetadata": {"RequestId": "947b0ba2-9561-5b7e-bbbc-569221b75cdd", "HTTPStatusCode": 200, "HTTPHeaders": {"x-amzn-requestid": "947b0ba2-9561-5b7e-bbbc-569221b75cdd", "content-type": "text/xml", "content-length": "294", "date": "Tue, 18 Jul 2023 15:41:10 GMT"}, "RetryAttempts": 0}}
Message: GAME RESULT league 52 team name 52 VS visitor 52 (0:0) MINUTES: 5
Response Payload: {"MessageId": "2667c9ab-5e80-533c-960c-949bea593e9c", "ResponseMetadata": {"RequestId": "36aa064f-f0cf-52f3-9026-5b7f4416ef53", "HTTPStatusCode": 200, "HTTPHeaders": {"x-amzn-requestid": "36aa064f-f0cf-52f3-9026-5b7f4416ef53", "content-type": "text/xml", "content-length": "294", "date": "Tue, 18 Jul 2023 15:41:13 GMT"}, "RetryAttempts": 0}}
Message: GAME RESULT league 52 team name 52 VS visitor 52 (0:0) MINUTES: 4
Response Payload: {"MessageId": "18ba3e2c-25a3-5503-a5fb-631c917f565b", "ResponseMetadata": {"RequestId": "94c437ca-d1e4-568e-a3bb-6afa653895d6", "HTTPStatusCode": 200, "HTTPHeaders": {"x-amzn-requestid": "94c437ca-d1e4-568e-a3bb-6afa653895d6", "content-type": "text/xml", "content-length": "294", "date": "Tue, 18 Jul 2023 15:41:16 GMT"}, "RetryAttempts": 0}}
Message: GAME RESULT league 52 team name 52 VS visitor 52 (0:0) MINUTES: 3
Response Payload: {"MessageId": "04eda2c1-78d0-545e-99b3-fff1cc0f4a8d", "ResponseMetadata": {"RequestId": "a35401cb-5463-5a79-bb00-5cfbe0bc24aa", "HTTPStatusCode": 200, "HTTPHeaders": {"x-amzn-requestid": "a35401cb-5463-5a79-bb00-5cfbe0bc24aa", "content-type": "text/xml", "content-length": "294", "date": "Tue, 18 Jul 2023 15:41:19 GMT"}, "RetryAttempts": 0}}
Message: GAME RESULT league 52 team name 52 VS visitor 52 (0:0) MINUTES: 2
Response Payload: {"MessageId": "c55f2ef4-8ee7-5b71-b3ff-73c43a056baa", "ResponseMetadata": {"RequestId": "14a2ae7d-3628-5528-94c8-f34d82ca05f3", "HTTPStatusCode": 200, "HTTPHeaders": {"x-amzn-requestid": "14a2ae7d-3628-5528-94c8-f34d82ca05f3", "content-type": "text/xml", "content-length": "294", "date": "Tue, 18 Jul 2023 15:41:21 GMT"}, "RetryAttempts": 0}}
Message: GAME RESULT league 52 team name 52 VS visitor 52 (0:0) MINUTES: 1
2023-07-18 15:52:59.943 INFO 35596 --- [nio-8081-exec-2] o.apache.coyote.http11.Http11Processor : Error parsing HTTP request header
Note: further occurrences of HTTP request parsing errors will be logged at DEBUG level.
```

i-008bc914d1b36111d (Sport-Service)

PublicIPs: 52.91.229.223 PrivateIPs: 172.31.94.13

EC2 Member service

The screenshot shows the Swagger UI interface for the EC2 Member service. At the top, the URL is `ec2-54-173-93-46.compute-1.amazonaws.com:8083/member/swagger-ui/index.html?configUrl=/member/api-docs/swagger-config`. The header includes the Swagger logo and the path `/member/api-docs`, with a green "Explore" button.

The main content area is titled "OpenAPI definition v0 OAS3". Below it, there's a "Servers" section with a dropdown set to `http://ec2-54-173-93-46.compute-1.amazonaws.com:8083/member - Generated server url`.

The API documentation is organized into several sections:

- Add Votes**: This is member api, it will add vote to current live game.
Method: POST /votes
- To test Api connection**: Just to check if api works
Method: GET /
- Get Votes**: This is public api, requires live gameid, to view live game comments.
Method: GET /votes
- Add Comments**: This member based api, requires accesstoken, user access to to add comment on live game
Method: POST /comments
- Get Comments**: This is public api anyone can access to view live game comments, but if no live game exists it will return empty array
Method: GET /comments

At the bottom, there's a "Schemas" section showing a "Comment" schema with a navigation arrow.

EC2 Auth service

The screenshot shows the Swagger UI interface for the EC2 Auth service. At the top, the URL is `ec2-44-202-164-100.compute-1.amazonaws.com:8080/auth/swagger-ui/index.html?configUrl=/auth/api-docs/swagger-config`. The header includes the Swagger logo and the path `/auth/api-docs`, with a green "Explore" button.

The main content area displays the "OpenAPI definition" for version v0, OAS3. It includes a "Servers" dropdown set to `http://ec2-44-202-164-100.compute-1.amazonaws.com:8080/auth - Generated server url`.

The API documentation is organized into sections:

- Login**: To check if login credentials are valid and it will return access token.
 - POST** /
- validatesAdmin**: To check if user provided token is admin user
 - PUT** /
- ValidateToken**: To check if access token is valid
 - GET** /
- auth-controller**
 - GET** /test
- Schemas**
 - AuthRequest** >

Auth service - RDS

The screenshot shows the Amazon RDS console interface. At the top, there's a navigation bar with the AWS logo, a services menu, a search bar, and account information for 'N. Virginia' and 'zaya @ aws-login-shujat'. Below the navigation is a sidebar with links like Dashboard, Databases, Query Editor, etc. The main content area shows a database named 'cs544' under the 'Databases' section. The 'Summary' tab is selected, displaying details such as DB identifier (cs544), CPU usage (3.29%), Status (Available), Class (db.t3.micro), Role (Instance), Current activity (20 Connections), Engine (MySQL Community), and Region & AZ (us-east-1b). Below the summary is a tabs section with 'Connectivity & security' (selected), Monitoring, Logs & events, Configuration, Maintenance & backups, and Tags. The 'Connectivity & security' tab shows endpoint and port details (Endpoint: cs544.c6wwj6rsn8el.us-east-1.rds.amazonaws.com, Port: 3306), networking (Availability Zone: us-east-1b, VPC: vpc-0659c130d3c0fb5e4), and security (VPC security groups: MySQL_SG (sg-0a842f7bd0e4e0f5d), Active). At the bottom, there are footer links for CloudShell, Feedback, Language, and a copyright notice: © 2023, Amazon Web Services, Inc. or its affiliates.

User Authentication:

Auth enables users to securely authenticate and access the system.

It supports various authentication methods, such as email and password.

Token-based Access:

The service generates and validates access tokens using JSON Web Tokens (JWT).

This ensures secure and stateless communication between clients and the server.

Admin Role:

Auth includes an admin role that grants additional privileges.

Admin users can manage system configurations, user roles, and permissions.

Creating SNS Topic

The screenshot shows the AWS SNS console in the us-east-1 region. A topic named "LiveScore" has been created. One subscription, which is an SQS queue, is listed. The "Endpoint" field shows the ARN of the SQS queue: "arn:aws:sqs:us-east-1:896553604990:Score". The "Status" of the subscription is "Confirmed".

ID	Endpoint	Status	Protocol
faffe1ae-378c-4ef7-81dc-7739215c4c01	arn:aws:sqs:us-east-1:896553604990:Score	Confirmed	SQS

Creating SQS Queue

The screenshot shows the AWS SQS console interface. At the top, the URL is `us-east-1.console.aws.amazon.com/sqs/v2/home?region=us-east-1#/queues/https%3A%2F%2Fsqs.us-east-1.am...`. The browser tab also displays this URL. The AWS logo and services navigation bar are visible. The main content area shows a queue named "Score".

Score Queue Details:

- Name: Score
- Type: Standard
- ARN: `arn:aws:sqs:us-east-1:896553604990:Score`
- Encryption: Amazon SQS key (SSE-SQS)
- URL: `https://sqs.us-east-1.amazonaws.com/896553604990/Score`
- Dead-letter queue: -

Buttons at the top right: Edit, Delete, Purge, Send and receive messages, Start DLQ redrive.

Navigation links below the details: SNS subscriptions, Lambda triggers, Dead-letter queue, Monitoring, Tagging, Access policy, Encryption, Dead-letter queue redrive tasks.

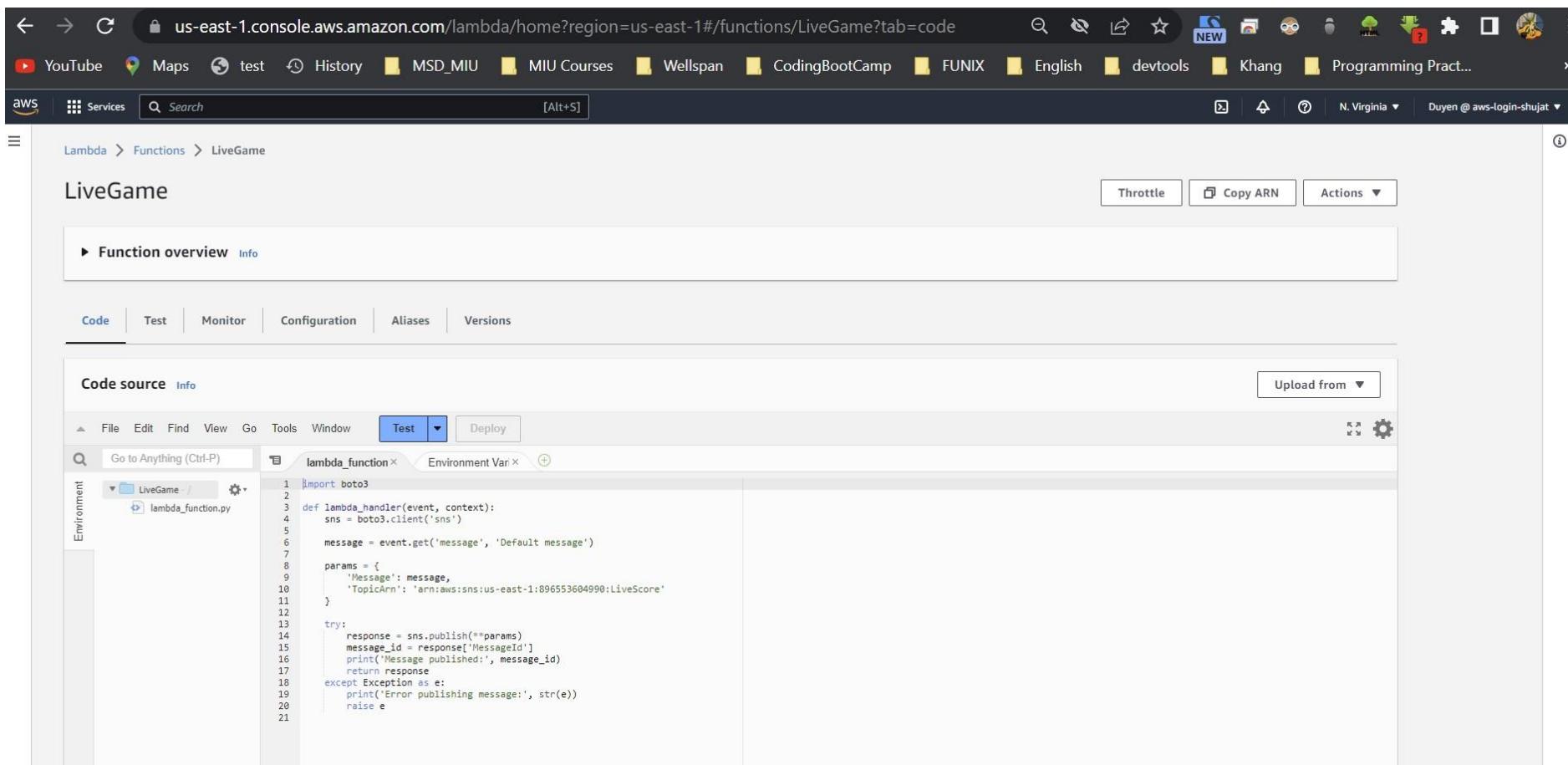
Subscription region: us-east-1

SNS subscriptions (1)

Subscription ARN	Topic ARN
<code>arn:aws:sns:us-east-1:896553604990:LiveScore:faffe1ae-378c-4ef7-81dc-7739215c4c01</code>	<code>arn:aws:sns:us-east-1:896553604990:LiveScore</code>

Actions: View in SNS, Delete, Subscribe to Amazon SNS topic.

Creating Lambda Function



Load Balancers & Target Group

The screenshot displays two AWS Management Console pages: 'Load balancers' and 'Target groups', both under the EC2 service.

Load balancers (3)

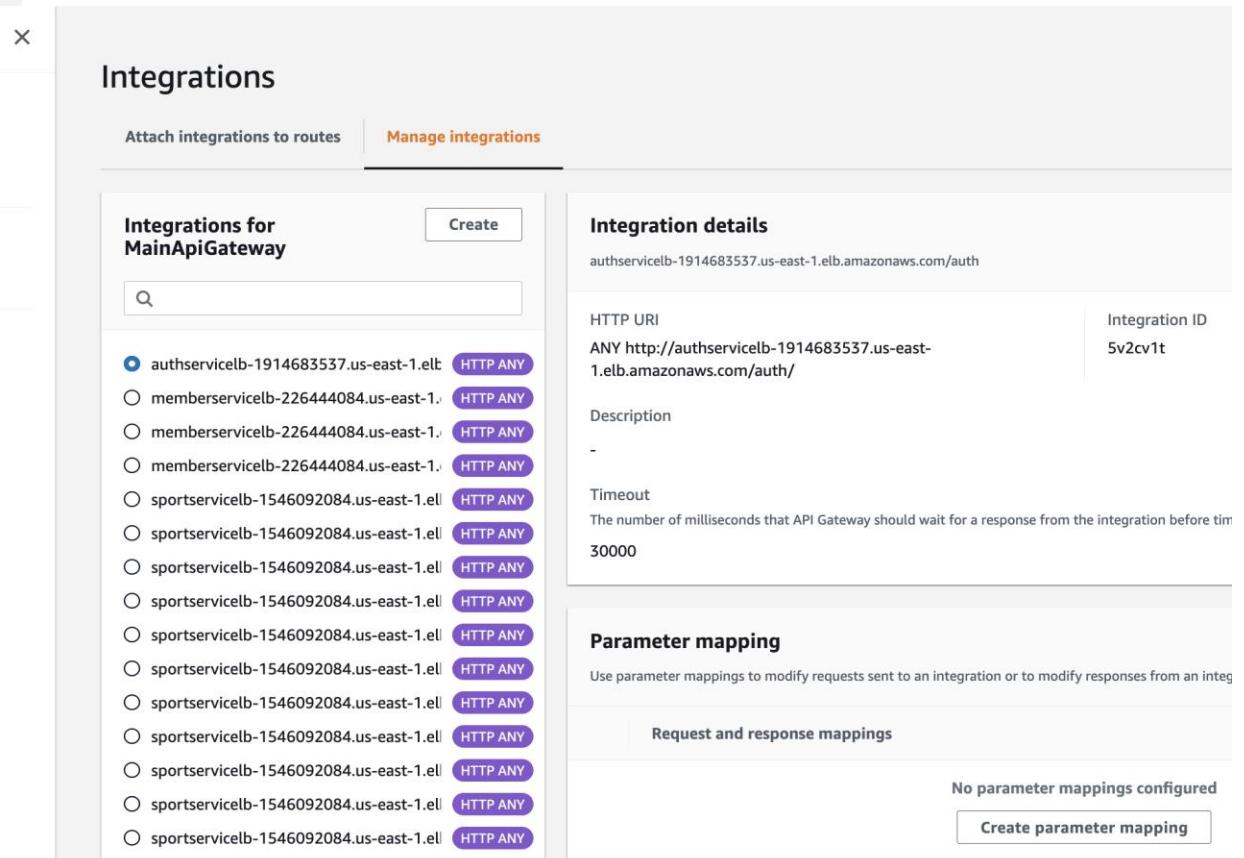
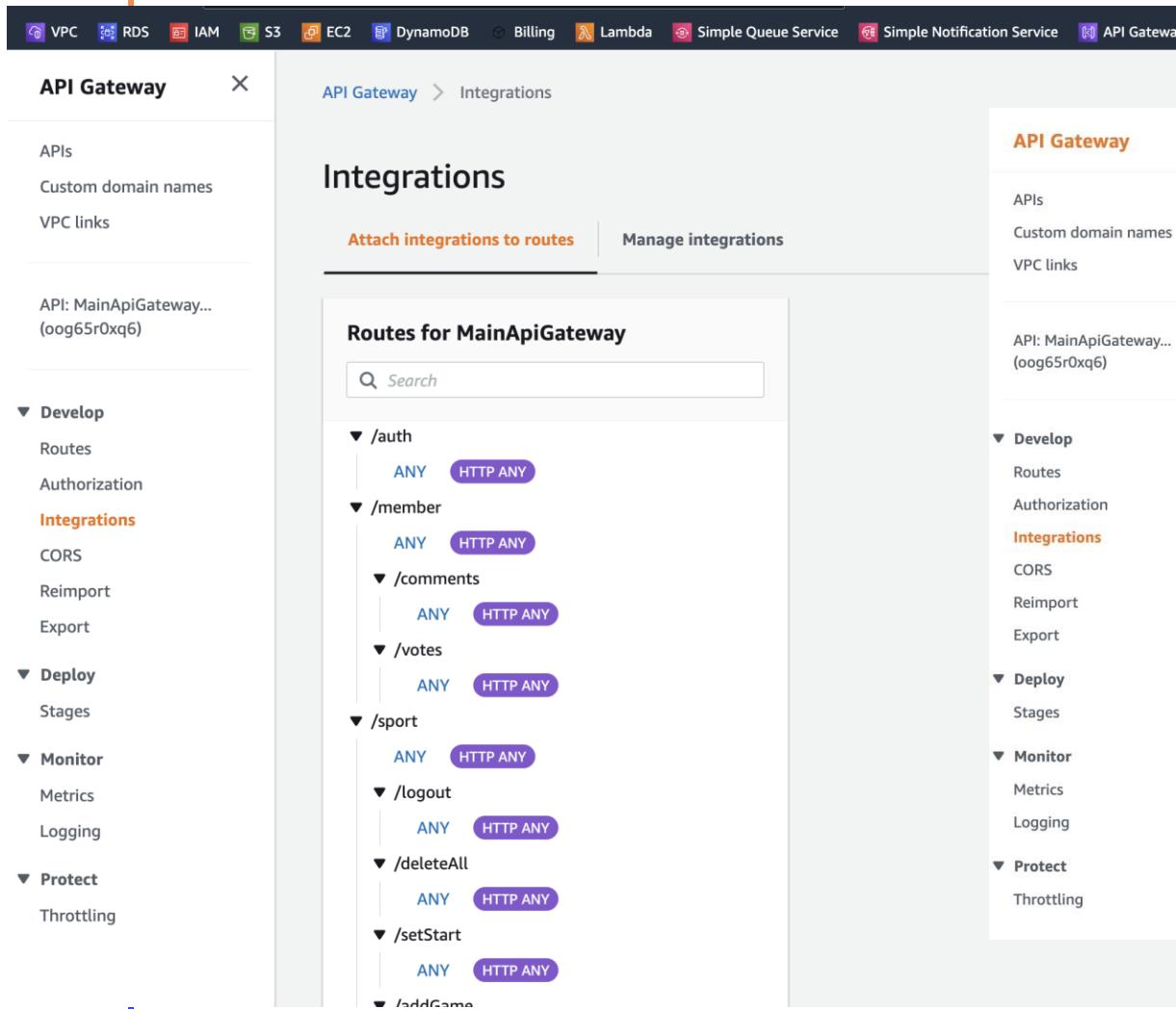
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Name	DNS name	State	VPC ID	Availability Zones	Type
MemberServiceLB	MemberServiceLB-2264...	Active	vpc-0659c130d3c0fb5e4	2 Availability Zones	application
SportServiceLB	SportServiceLB-154609...	Active	vpc-0659c130d3c0fb5e4	3 Availability Zones	application
AuthServiceLB	AuthServiceLB-1914683...	Active	vpc-0659c130d3c0fb5e4	2 Availability Zones	application

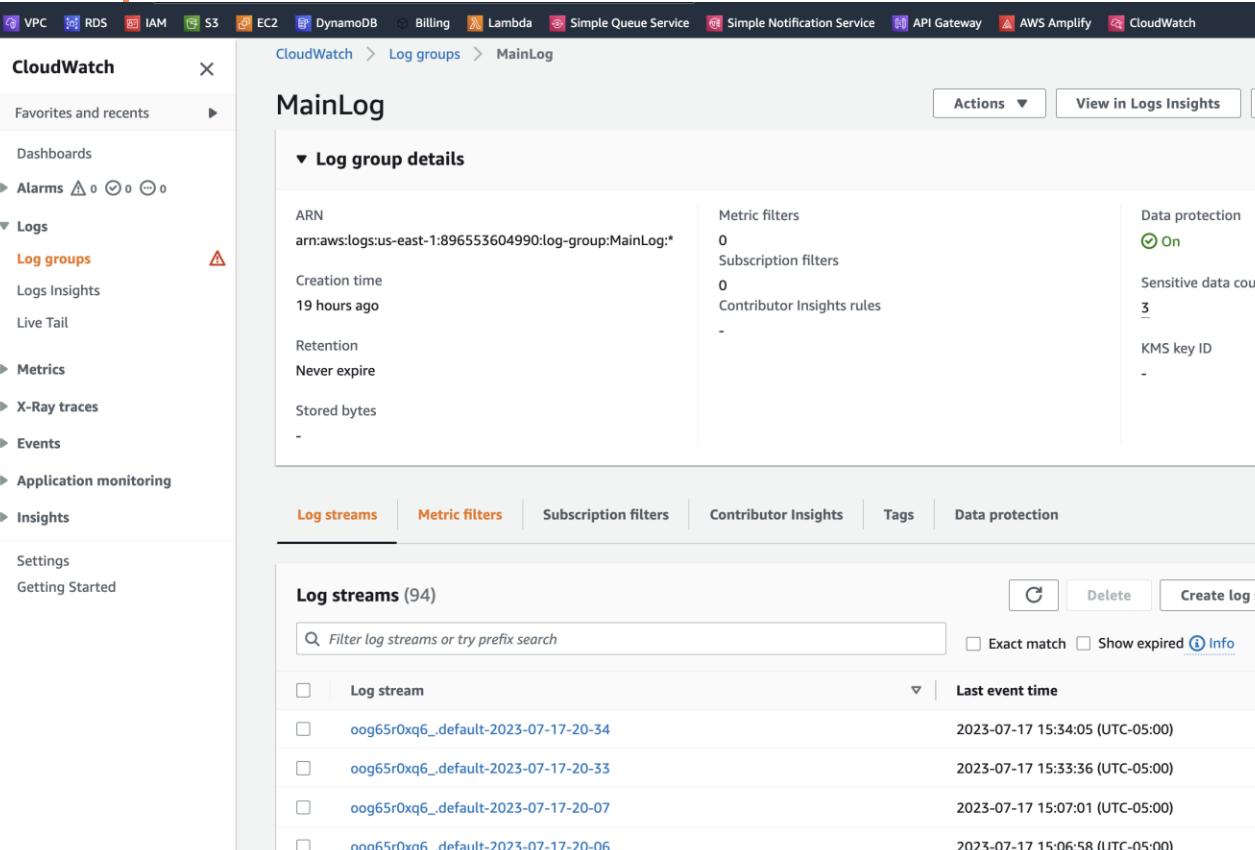
Target groups (3) Info

Name	ARN	Port	Protocol	Target type	Load balancer
AuthServiceTG	arn:aws:elasticloadbalanci...	8080	HTTP	Instance	AuthServiceLB
MemberServiceTG	arn:aws:elasticloadbalanci...	8083	HTTP	Instance	MemberServiceLB
SportServiceTG	arn:aws:elasticloadbalanci...	8081	HTTP	Instance	SportServiceLB

Api Gateway & Api Integrations



CloudWatch & Logging S3



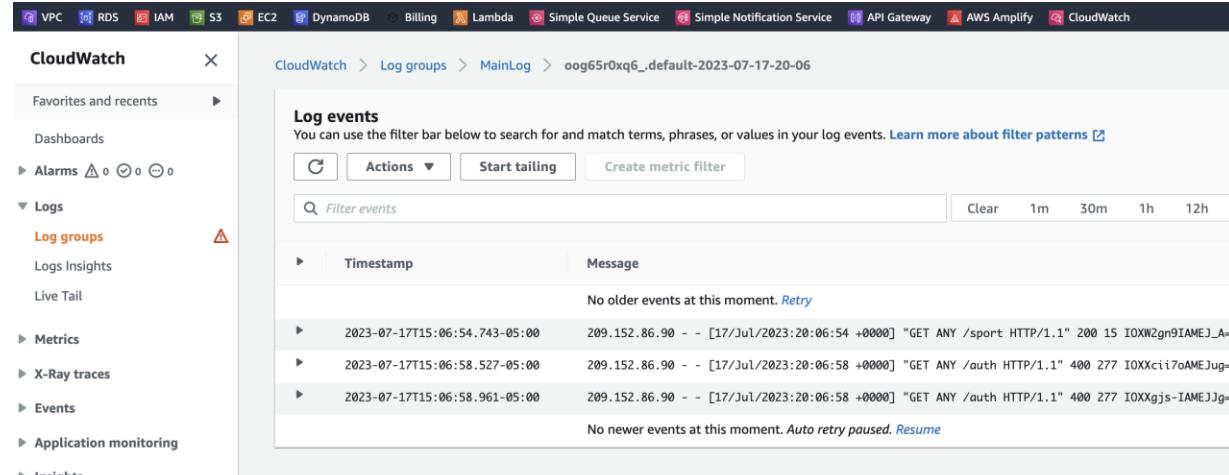
The screenshot shows the AWS CloudWatch Logs interface. On the left, a sidebar navigation includes VPC, RDS, IAM, S3, EC2, DynamoDB, Billing, Lambda, Simple Queue Service, Simple Notification Service, API Gateway, AWS Amplify, and CloudWatch. Under the CloudWatch section, the 'Logs' category is selected, showing 'Log groups' (MainLog) and other options like Logs Insights, Live Tail, Metrics, X-Ray traces, Events, Application monitoring, and Insights.

MainLog Details:

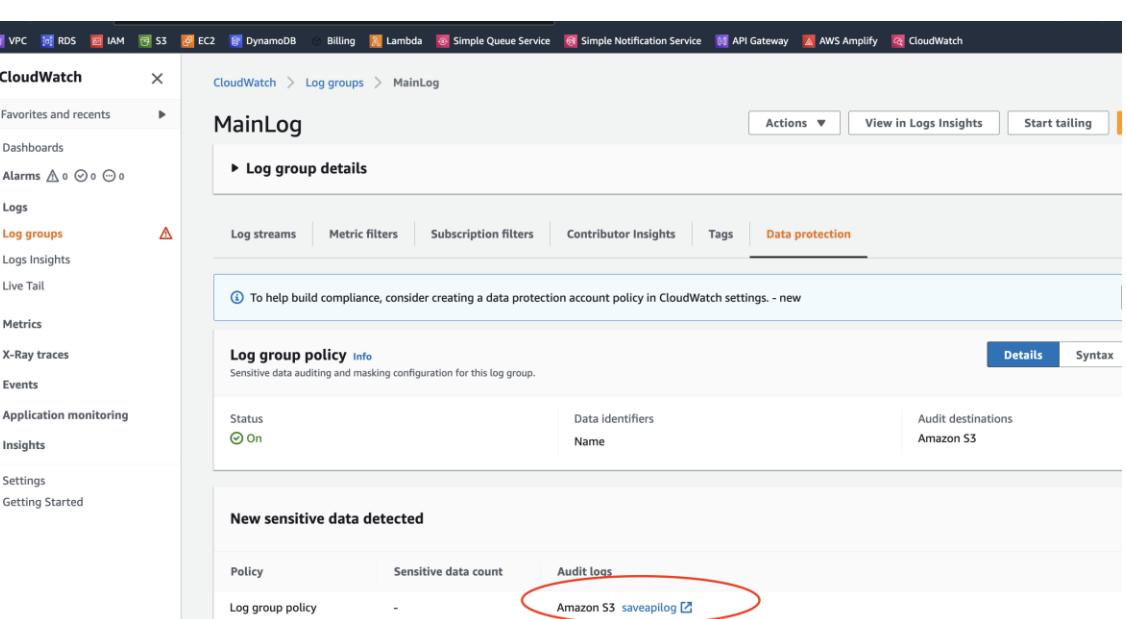
- ARN: arn:aws:logs:us-east-1:896553604990:log-group>MainLog*
- Creation time: 19 hours ago
- Retention: Never expire
- Stored bytes: -
- Metric filters: 0
- Subscription filters: 0
- Contributor Insights rules: -
- Data protection: On (green checkmark)
- Sensitive data count: 3
- KMS key ID: -

Log streams (94):

Log stream	Last event time
oog65r0xq6_default-2023-07-17-20-34	2023-07-17 15:34:05 (UTC-05:00)
oog65r0xq6_default-2023-07-17-20-33	2023-07-17 15:33:36 (UTC-05:00)
oog65r0xq6_default-2023-07-17-20-07	2023-07-17 15:07:01 (UTC-05:00)
oog65r0xq6_default-2023-07-17-20-06	2023-07-17 15:06:58 (UTC-05:00)

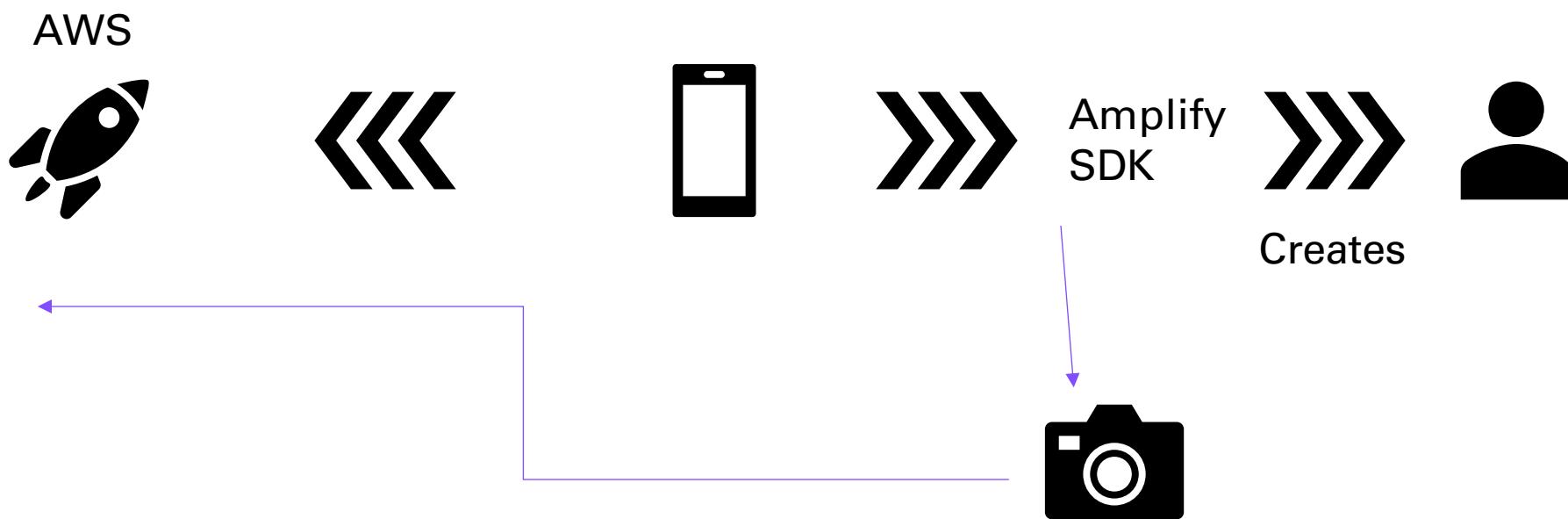


The screenshot shows the AWS CloudWatch Log Events interface for the log group 'oog65r0xq6_default-2023-07-17-20-06'. It displays log events with columns for Timestamp and Message. The first event is a placeholder message: "No older events at this moment. Retry". Subsequent events show log entries from July 17, 2023, at 15:06:58 UTC-05:00, such as "GET ANY /sport HTTP/1.1" and "GET ANY /auth HTTP/1.1".



The screenshot shows the AWS CloudWatch Log Group Policy interface for the 'MainLog' group. It highlights a 'New sensitive data detected' entry. The 'Audit logs' section shows a policy named 'Log group policy' with an audit destination of 'Amazon S3 saveapilog'. A red oval surrounds the 'Amazon S3 saveapilog' link.

AWS Amplify And Mobile



AWS Amplify

sandbox.amplifyapp.com/getting-started

Amplify Studio

Home Try without AWS account Data UI library Sign in to AWS to use Host my app Authentication File storage Functions GraphQL API Analytics Predictions Interactions Notifications

Share Getting started progress 0% aws Unlock all of Amplify Deploy to AWS

Quick start Build a Blog with Android (Kotlin) Get started

① Data model ② Test in your app ③ Deploy to AWS AWS account required

Build an app from scratch Select framework Android (Kotlin)

Create data model Building from scratch

The screenshot shows the AWS Amplify Studio interface. At the top, there's a navigation bar with icons for back, forward, search, and other browser functions. Below the bar, the title 'Amplify Studio' is displayed next to a logo. To the right are 'Share', 'Getting started progress' (0%), 'aws Unlock all of Amplify', and a 'Deploy to AWS' button. On the left, a sidebar lists various services: Home, Try without AWS account, Data, UI library, Sign in to AWS to use, Host my app, Authentication, File storage, Functions, GraphQL API, Analytics, Predictions, Interactions, and Notifications. The main area has a dark background with a decorative starry pattern. A central 'Quick start' card is open, showing options to 'Build a Blog' using 'Android (Kotlin)'. Below this, three steps are listed: 'Data model', 'Test in your app', and 'Deploy to AWS', with a note that 'AWS account required'. At the bottom, there's a large call-to-action for 'Build an app from scratch' with a 'Select framework' dropdown set to 'Android (Kotlin)'.

AWS Amplify

The screenshot shows the AWS Amplify service page within the AWS Management Console. The top navigation bar includes the AWS logo, a services menu, a search bar, and user account information for "Shujat ali" in the "Ohio" region. Below the navigation is a horizontal menu bar with icons for VPC, RDS, IAM, S3, EC2, DynamoDB, Billing, Lambda, Simple Queue Service, Simple Notification Service, API Gateway, AWS Amplify (which is selected), and CloudWatch.

The main content area displays the AWS Amplify landing page. It features a large central icon with three yellow triangles forming a circle, followed by the text "AWS Amplify" and the tagline "Fastest, easiest way to develop mobile and web apps that scale." A prominent orange "GET STARTED" button is centered below this. At the bottom, it lists supported technologies: React, Vue, JavaScript, Node.js, iOS, Android, and Flutter. A descriptive paragraph explains that AWS Amplify provides tools for building secure, scalable full-stack applications. A dark callout box at the bottom contains the text "AWS Amplify: Fastest, Easiest W..." followed by a vertical ellipsis.

AWS Amplify

Amplify Studio SoccerGame > amplifyapp ▾ Local setup instructions  AW

Home Manage Content User management File browser

UI Library NEW Set up Data Authentication Storage Functions GraphQL API REST API

Documentation Support Feedback

User management
Create, view, and manage users and groups for your application.

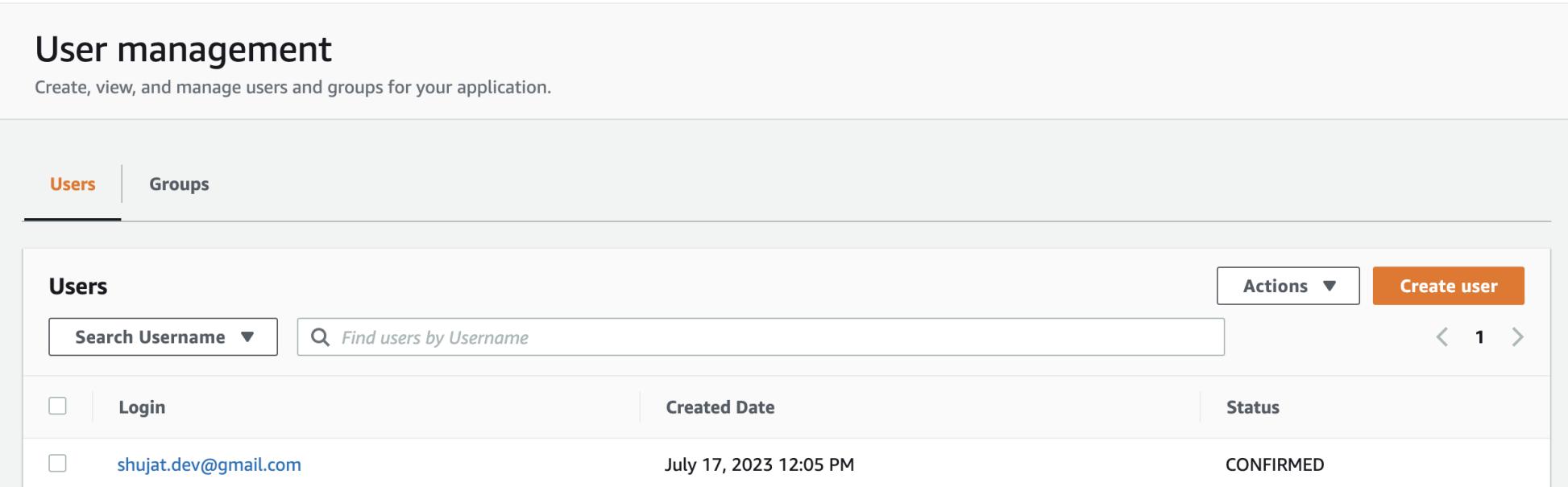
Users Groups

Users

Search Username ▾ Find users by Username Actions ▾ Create user

<input type="checkbox"/>	Login	Created Date	Status
<input type="checkbox"/>	shujat.dev@gmail.com	July 17, 2023 12:05 PM	CONFIRMED

< 1 >



AWS Amplify

AWS Services Search [Option+S] N. Virginia Shujat ali

VPC RDS IAM S3 EC2 DynamoDB Billing Lambda Simple Queue Service Simple Notification Service API Gateway AWS Amplify CloudWatch

AWS Amplify All apps > SoccerGame **SoccerGame** Actions ▾

All apps SoccerGame

App settings General Amplify Studio settings

Documentation Support

Learn how to get the most out of Amplify Studio X

Hosting environments Backend environments

This tab lists all backend environments. Each backend environment is a container for all of the cloud capabilities added to your app such as API, auth, and storage.

amplifyapp Continuous deploys not set up. Actions ▾



Deployment status Deployment completed 17/07/2023, 12:00:22

Categories added Authentication Storage

Launch Studio

AWS Amplify

Amplify Studio

SoccerGame > amplifyapp ▾

Local setup instructions  

- Home
- Manage
 - Content
 - User management
 - File browser
- Design
 - UI Library NEW
- Set up
 - Data
 - Authentication
 - Storage Storage
 - Functions
 - GraphQL API
 - REST API
- Documentation 
- Support 
- Feedback 

File storage

 **Deployed** Manage storage resources for file storage (images, audio, video etc.) and data storage backend by Amazon S3.

S3 bucket information

[Unlink S3 storage bucket](#)

S3 bucket name

soccerbucket115925-amplifyapp

S3 bucket information

<s3://soccerbucket115925-amplifyapp> 

Set authorization rules

Signed-in users

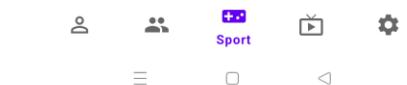
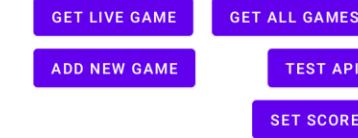
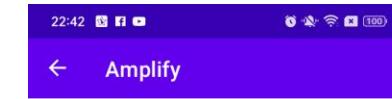
Upload View Delete

Guest users - *optional*

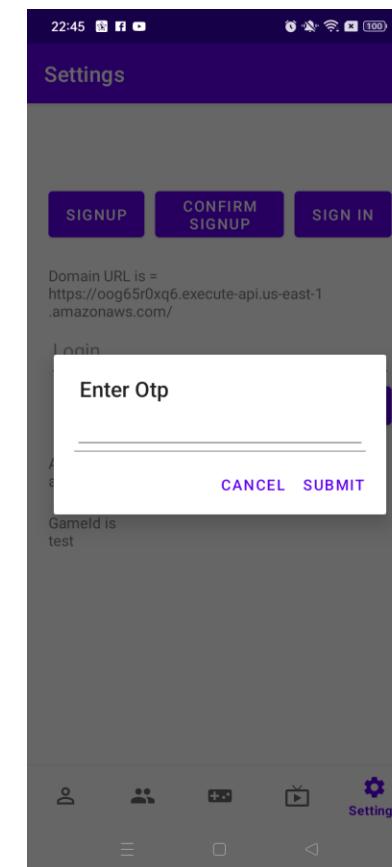
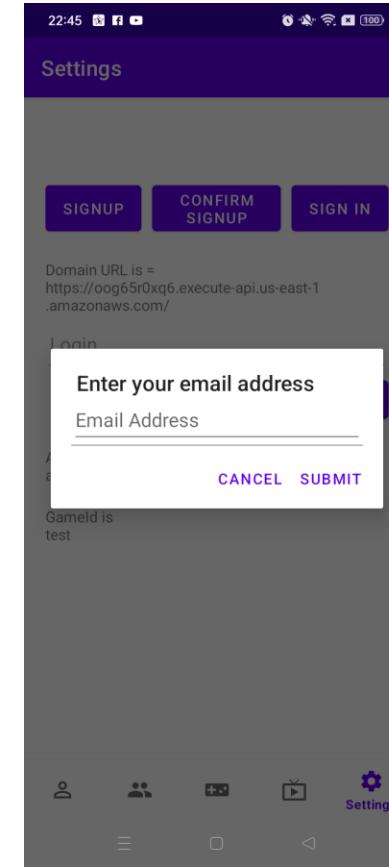
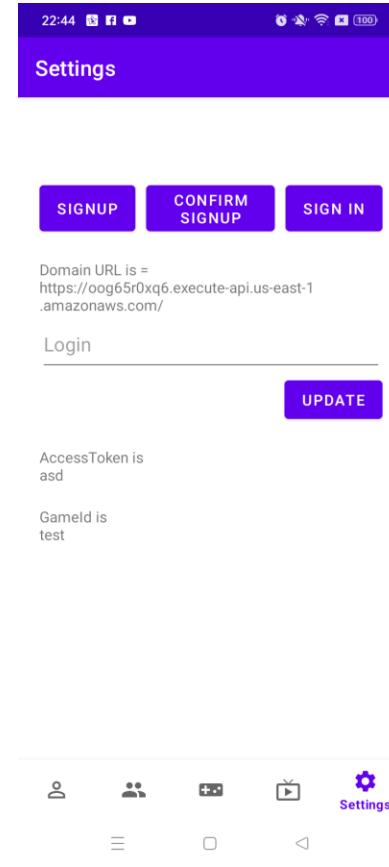
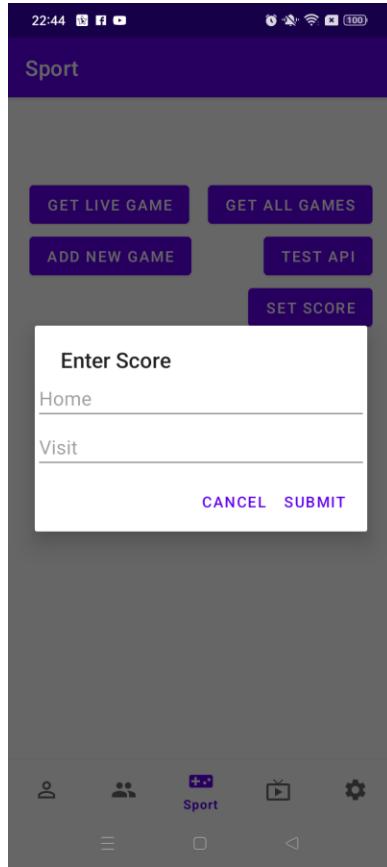
Upload View Delete

[Deploy](#)

Android Mobile Client



Android Mobile Client





+



O



.



THANK YOU